Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Please cancel claims 1-42 and 49-51 without prejudice.

Claims 1-42 (canceled)

43. (original): A method of transfer control operation comprising the steps of:

detecting a transfer controller reset event;

comparing a transfer program counter value with a wait program counter value;

updating either one of the transfer program counter value or the wait program counter value;

determining that the transfer program counter value and the wait program counter value are not equal; and

placing a transfer controller in a fetch state upon said determination.

44. (original): The method of claim 43 further comprising the steps of fetching and decoding an instruction word.

45. (original): The method of claim 44 wherein if an instruction comprises multiple words, repeating the step of decoding until all the multiple words are decoded.

46. (original): The method of claim 44 further comprising the step of incrementing the transfer counter by one each time a word is fetched.

47. (original): The method of claim 44 further comprising the steps of determining that a fetched instruction is a control type instruction; transitioning to an execute control state; and performing an action specified by the fetched instruction.

48. (original): The method of claim 44 further comprising the steps of executing a wait type instruction and causing a transfer controller to transition to a wait state.

Claims 49-51 (canceled)

52. (original): A multiprocessor DMA system comprising: at least a first processor and a second processor to carry out DMA-to-DMA transfers between DMA controllers or I/O devices.

53. (original): The multiprocessor DMA system of claim 52 wherein said processors employ a push model DMA-to-DMA transfer, each processor further comprising a transfer controller that is reading a data source which acts as a system data bus (SDB) master and writes data to an SDB slave address range of another transfer controller that is writing data to a destination memory.

54. (original): The multiprocessor DMA system of claim 52 wherein said processors employ a pull model DMA-to-DMA transfer, each processor further comprising a transfer controller that is writing data to its destination memory which acts as an SDB master and reads data from an SDB slave address range of another transfer controller that is reading data from a source memory.

Please add new claims as follows:

55. (new): A direct memory access (DMA) controller disposed within a processing system, the DMA controller connected to a system data bus (SDB), the system data bus carrying data to a processor connected to the system data bus, the DMA controller further connected to a core memory within the processing system, the DMA controller operable to read from or write to

the core memory, the DMA controller operable to read from or write to the SDB, the DMA controller comprising:

a first transfer controller running in its own thread of execution independent of another processor disposed with the processing system to carry out data transfers between the system data bus and the core memory, the first transfer controller having a data queue, a first execution unit for transferring data between the core memory and the data queue, and a second execution unit for transferring data between the SDB and the data queue, the second execution unit having at least active and deactivate states;

a first outbound transfer instruction, when executed by the first execution unit, causing the first execution unit to transfer data from the core memory to the data queue; and

a second outbound transfer instruction, when executed by the second execution unit in the active state, causing the second execution unit to transfer data from the data queue to the SDB.

56. (new): The DMA controller of claim 55 further comprising:

a first inbound transfer instruction, when executed by the second execution unit in the active state, causing the second execution unit to transfer data from the SDB to the data queue; and

a second inbound transfer instruction, when executed by the first execution unit, causing the first execution unit to transfer data from the data queue to the core memory.

57. (new): The DMA controller of claim 56 wherein the DMA controller transfers data to core memory from a second DMA controller connected to the SDB, the second execution unit executing the first inbound transfer instruction, the first execution unit executing the second inbound transfer instruction;

58. (new): The DMA controller of claim 56

wherein the DMA controller transfers data to core memory from a second DMA controller connected to the SDB;

wherein the first transfer controller further comprises a slave address, the second DMA controller writes the data to the slave address bypassing the second execution unit in the deactive state and queuing the data directly to the data queue; and

wherein the first execution unit executes the second inbound transfer instruction to transfer the data from the data queue to the core memory.

59. (new): The DMA controller of claim 55

wherein the DMA controller transfers data to a second DMA controller connected to the SDB;

wherein the first transfer controller further comprises a slave address;

wherein the second execution unit is deactivated; and

wherein the first execution unit executes the first outbound transfer instruction to transfer from the core memory to the data queue, the second DMA controller retrieves the data from the data queue by reading the slave address;

60. (new): The DMA controller of claim 56 further comprising:

a second transfer controller connected to the core memory over an independent data path and the SDB, the second transfer controller controlling concurrent data transfer in a first direction between the core memory and the SDB, the first transfer controller controlling data transfer in a second direction between the core memory and the SDB, the first direction is opposite to the second direction.

61. (new): A method for transferring data by a DMA controller disposed within a processing system having core memory and a system data bus (SDB), the DMA controller

having a transfer controller, the transfer controller having first execution unit, a second execution unit, and a data queue, the method comprising:

operating the transfer controller in its own thread of execution independent of another processor disposed within the processing system;

executing a first outbound transfer instruction by the first execution unit to transfer data from the core memory to the data queue;

activating the second execution unit; and

executing a second outbound transfer instruction by the second execution unit to transfer data from the data queue to the SDB.

62. (new): The method of claim 61 wherein the method transfers data to core memory from a second DMA controller connected to the SDB, the method further comprising:

executing a first inbound transfer instruction by a second execution unit to transfer data from the SDB to the data queue; and

executing a second inbound transfer instruction by a first execution unit to transfer data from the data queue to the core memory.

63. (new): The method of claim 61 wherein the method transfers data to core memory from a second DMA controller connected to the SDB, the method further comprising:

deactivating the second execution unit;

writing data by the second DMA controller to a slave address in the transfer controller to queue the data directly to the data queue; and

executing the second inbound transfer instruction to transfer the data from the data queue to the core memory.

64. (new): The DMA controller of claim 61 wherein the method transfers data to a second DMA controller through the SDB, the method comprising:

deactivating the second execution unit;

executing the first outbound transfer instruction to transfer from the core memory to the data queue; and

reading a slave address to retrieve the data from the data queue over the SDB.